11-711: Notes on Hidden Markov Model

Fall 2017

1 Hidden Markov Model

Hidden Markov Model (HMM) is a parameterized distribution for sequences of observations. An intuitive way to explain HMM is to go through an example. Suppose that Taylor hears (a.k.a. observes) a sequence of $T$ sounds $o_1, o_2, \ldots, o_T$ and he wants to reason something about this sequence. He makes the assumption that the sequence of sounds that he heard depends on a sequence of $T$ words $s_1, s_2, \ldots, s_T$, which he never gets to see and which is why they are called the hidden states. HMM gives Taylor a method which, under certain assumptions, allows him to assign appropriate probabilities to sound sequences $O$’s and word sequences $S$’s and to make reasonable deductions about them. Figure 1 illustrates an HMM with $T = 5$.

![Figure 1: An illustration of a Hidden Markov Model with 5 steps](image)

HMM makes several assumptions about the data it models:

1. The observations $o_1, o_2, \ldots, o_T$ come from a known, finite sets $V$, called the observation space.
2. The hidden states $s_1, s_2, \ldots, s_T$ come from a known, finite sets $Q$, called the hidden state space.
3. The HMM assigns a probability to any given sequence $s_1, s_2, \ldots, s_T$. The chain rule says that

$$P(s_1, s_2, \ldots, s_T) = \prod_{t=1}^{T} P(s_t | s_{0}, s_1, \ldots, s_{t-1}) = \prod_{t=1}^{T} P(s_t | s_{<t}) \quad (1.1)$$

Here, we are being a little sloppy in terms of notation by assuming that there is a starting state $s_0$, just so that the indexing is valid. In practice, such state $s_0$ is sometimes called the starting state, and one can assign a probability distribution to it too.

HMM goes further than Equation 1.1 by making the assumption that

$$P(s_t | s_{<t}) = P(s_t | s_{t-1}) \quad (1.2)$$
This is called the **Markov assumption**. With this assumption, Equation 1.1 can be rewritten as

\[ P(s_1, s_2, \ldots, s_T) = \prod_{t=1}^{T} P(s_t|s_{t-1}) \tag{1.3} \]

Note that the Markov assumption generally does not hold. In our example, sequences of words usually have dependencies that go back longer than one immediate step.

4. Each observation \( o_t \) only depends on the immediate hidden state \( s_t \). Formally

\[ P(o_t|o_{<t}, s_{<t}) = P(o_t|s_t) \tag{1.4} \]

This is called the **independence assumption**.

Under the Markov assumption and the independence assumption, an HMM only needs two set of parameters to model \( P(s_t|s_{t-1}) \) and \( P(o_t|s_t) \). We call these, respectively, the **transition probabilities** and the **emission probabilities**. These probabilities are denoted by matrices \( A \) and \( B \), each of whose row is a valid probability distribution, i.e. non-negative numbers that sum up to 1. Table 1 summarizes the components of a general HMM.

<table>
<thead>
<tr>
<th>Name</th>
<th>Notation</th>
<th>Meaning/Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>State space</td>
<td>( Q = {q_1, q_2, \ldots, q_N} )</td>
<td>Set of ( N ) states</td>
</tr>
<tr>
<td>Observation space</td>
<td>( V = {w_1, w_2, \ldots, w_V} )</td>
<td>Set of ( V ) states</td>
</tr>
<tr>
<td>State sequence</td>
<td>( S = {s_1, s_2, \ldots, s_T} )</td>
<td>Sequence of ( T ) steps. ( s_i \in Q )</td>
</tr>
<tr>
<td>Observation sequence</td>
<td>( O = {o_1, o_2, \ldots, o_T} )</td>
<td>Sequence of ( T ) steps. ( o_t \in V )</td>
</tr>
<tr>
<td>Transition probs</td>
<td>( A \in \mathbb{R}^{N \times N} )</td>
<td>Each row is a valid distribution</td>
</tr>
<tr>
<td>Emission probs</td>
<td>( B \in \mathbb{R}^{N \times V} )</td>
<td>Each row is a valid distribution</td>
</tr>
</tbody>
</table>

Table 1: Components of a Hidden Markov Model.

There are three important questions regarding HMM.

1. **Computing the Likelihood.** Given \( A, B \) and a sequence of observations \( O \). Compute \( P(O|A, B) \).

2. **Finding the Hidden Sequence.** Given \( A, B \) and a sequence of observation \( O \). Find the hidden sequence \( S \) that is most likely to generate \( O \), that is to find

\[ S^* = \arg \max_S P(O|S, A, B) \tag{1.5} \]

3. **Estimating the Parameters.** Given a sequence of observation \( O \). Estimate the transition and emission probabilities that are most likely to give \( O \). That is to find

\[ A^*, B^* = \arg \max_{A,B} P(O|A, B) \tag{1.6} \]

In the following sections, we provide the algorithms to address each of them.

## 2 Computing the Likelihood: Forward-Pass Algorithm

Given a sequence \( O = (o_1, o_2, \ldots, o_T) \) and known transition and emission probabilities matrices \( A, B \). We want to find \( P(O|A, B) \). We use dynamic programming. Specifically, for each step \( t = 1, 2, \ldots, T \) and each hidden state \( s \), we will compute and cache the following value

\[ f[t, s] = P(o_1, o_2, \ldots, o_t, s_t = s|A, B) \tag{2.1} \]
If we know \( f[t,s] \), we can compute \( P(O|A,B) \) as follows

\[
P(o_1, o_2, ..., o_t|A,B) = \sum_s P(o_1, o_2, ..., o_t, s|A,B) \cdot P(o_t|s_t = s)
\] (2.2)

\[
= \sum_s f[t,s] \cdot B(s,o_t)
\] (2.3)

Now we can compute \( f[t,s] \) from previous values \( f[t-1,s] \). Specifically

\[
f[t,s] = P(o_1, o_2, ..., o_t, s_t = s|A,B)
\] (2.4)

\[
= \sum_{s'} P(o_1, o_2, ..., o_{t-1}, s_{t-1} = s'|A,B) \cdot P(s_t = s|s_{t-1} = s') P(o_t|s)
\] (2.5)

\[
= \sum_{s'} f[t-1,s'] \cdot A(s',s) B(s,o_t)
\] (2.6)

Putting together Equation 2.3 and Equation 2.6, we have the so-called Forward-Pass algorithm to compute the likelihood in an HMM.

(1) Initialize:

- For each hidden state \( s \):
  \[
f[1,s] = P(o_1, s_1 = s|A,B) \leftarrow B(s,o_1) \cdot A(s_0,s)
\] (2.7)

(2) For \( t = 2 \) to \( T \):

- For each hidden state \( s \):
  \[
f[t,s] \leftarrow \sum_{s'} f[t-1,s'] \cdot A(s',s) B(s,o_t)
\] (2.8)

(3) Finally:

\[
P(o_1, o_2, ..., o_T|A,B) \leftarrow \sum_s f[T,s] B(s,o_T)
\] (2.9)

Its complexity is \( O(N^2 \cdot T) \) where \( N \) is the size of the hidden state space.

### 3 Finding the Hidden Sequence: Viterbi Algorithm

Given a sequence \( O = (o_1, o_2, ..., o_T) \) and known transition and emission probabilities matrices \( A, B \). For each hidden sequence \( S \), the joint probability that both \( O \) and \( S \) happen is

\[
P(O,S|A,B) = P(O|S,A,B) \cdot P(S|A,B)
\] (3.1)

\[
= \prod_{i=1}^{T} P(o_i|s_i) \cdot \prod_{i=1}^{T} P(s_i|s_{i-1})
\] (3.2)

\[
= \prod_{i=1}^{T} B(s_i, o_i) \cdot \prod_{i=1}^{T} A(s_{i-1}, s_i)
\] (3.3)
We want to find the sequence $S$ that maximizes the quantity above, that is

$$S^* = \arg\max_S P(O, S|A, B)$$  \hspace{1cm} (3.4)

It turns out that a dynamic programming programming algorithm that is almost identical to the Forward-Pass algorithm in Section 2 is applicable for the task of finding $S$ in Equation 3.4. That is the Viterbi algorithm \cite{Viterbi, 1967}.

We define

$$g[t, s] = \max_{s_1, \ldots, s_{t-1}} P(o_1, o_2, \ldots, o_t, s_t = s|A, B)$$  \hspace{1cm} (3.5)

The key observation is that if we know all values of $g[t, s]$, then we can compute the maximum joint probability $P(O, S|A, B)$ as follows

$$\max_{s_1, \ldots, s_t} P(o_1, o_2, \ldots, o_t, s_t = s|A, B) = \max_s P(o_1, o_2, \ldots, o_t, s_t = s|A, B) \cdot P(o_t|s_t = s)$$  \hspace{1cm} (3.6)

$$= \max_s g[t, s] \cdot B(s, o_t)$$  \hspace{1cm} (3.7)

Please note the similarity between Equations 3.6-3.7 to Equations 2.2-2.3, where the only difference is the change from summation to max. The reason that Viterbi’s algorithm is almost identical to Forward-Pass algorithm is that max has the same distributing property as summation, i.e. one can “factor” our the same term.

We can compute $g[t, s]$ similar to the way we compute $f[t, s]$ in Equation 2.6. Specifically

$$g[t, s] = \max_{s_1, \ldots, s_{t-1}} P(o_1, o_2, \ldots, o_t, s_t = s|A, B)$$  \hspace{1cm} (3.8)

$$= \max_s P(o_1, o_2, \ldots, o_t, s_{t-1} = s'|A, B) \cdot P(s_t = s|s_{t-1} = s')P(o_t|s)$$  \hspace{1cm} (3.9)

$$= \max_{s'} g[t-1, s'] \cdot A(s', s)B(s, o_t)$$  \hspace{1cm} (3.10)

Finally, note that in Equation 3.4, we wanted an $\arg\max_S P(O, S|A, B)$, not the $\max_S P(O, S|A, B)$. The general idea to go from max(·) to argmax(·) in a dynamic programming algorithm is to create a trace-back array. In Equation 3.10, whenever you know which $s'$ leads to the largest value for $g[t-1, s'] \cdot A(s', s)B(s, o_t)$, you can store that $s'$ in an array $h[t, s]$. After you finish with $g[T, s]$ as in Equation 3.7, you can simply follow $h[t, s]$ to trace back the sequence of hidden states that leads to the maximum value of $P(O, S|A, B)$, which gives you the argmax. Putting everything together, we have the Viterbi’s algorithm

\begin{enumerate}
  \item \textbf{Initialize:}
    \begin{itemize}
      \item For each hidden state $s$:
        $$g[1, s] \leftarrow B(s, o_1) \cdot A(s_0, s)$$  \hspace{1cm} (3.11)
    \end{itemize}
  \item \textbf{For $t = 2$ to $T$:}
    \begin{itemize}
      \item For each hidden state $s$:
        $$g[t, s] \leftarrow \max_{s'} g[t - 1, s'] \cdot A(s', s)B(s, o_t)$$  \hspace{1cm} (3.12)
        $$h[t, s] \leftarrow \arg\max_{s'} g[t - 1, s'] \cdot A(s', s)B(s, o_t)$$  \hspace{1cm} (3.13)
    \end{itemize}
\end{enumerate}
4 Estimating the Parameters: Baum-Welch Algorithm

In Section 2 and Section 3, we have seen that if we know the transition probabilities \( A(s,s') \) and the emission probabilities \( B(s,o) \), then we can efficiently perform the queries on an HMM. In this section, we are concerned with the problem of estimating \( A \) and \( B \), given a single sequence of observations \( O = (o_1, o_2, \ldots, o_T) \).

The algorithm for this task is the Baum-Welch algorithm [Baum, 1972], an instance of a procedure called Expectation-Maximization (EM) [Dempster et al., 1977]. In Baum-Welch algorithm, one starts by randomly initializing the values for \( A \) and \( B \), then repeatedly updating them. Each updating iteration has two steps, the Expectation step (E-step) and the Maximization step (M-step).

**E-step.** The E-step assumes that we know \( A, B \) and computes the following quantities

\[
\begin{align*}
\gamma[t,s] &= P(s_t = s|O,A,B) \tag{4.1} \\
\xi[t,s',s] &= P(s_{t-1} = s',s_t = s|O,A,B) \tag{4.2}
\end{align*}
\]

Intuitively, \( \gamma[t,s] \) counts how many times does the \( t \)-th states of the hidden sequence equals \( s \) and \( \xi[t,s,s'] \) counts how many times the duo \( (s',s) \) happens at the \( (t-1) \)-th step and the \( t \)-th step in the hidden sequence, both up to normalization by a partition function. That is why \( \gamma[t,s] \) and \( \xi[t,s,s'] \) are sometimes referred to as the pseudo counts.

We now show how to compute \( \gamma[t,s] \) and \( \xi[t,s',s] \). Since \( \gamma[t,s] \) and \( \xi[t,s',s] \) are probabilities conditioned on \( O \), we compute via the partial joint probabilities

\[
\begin{align*}
\alpha[t,s] &= P(o_1, o_2, \ldots, o_t, s_t = s|A,B) \tag{4.3} \\
\beta[t,s] &= P(o_{t+1}, o_{t+2}, \ldots, o_T|s_t = s, A,B) \tag{4.4}
\end{align*}
\]

If one knows all \( \alpha[t,s] \) and \( \beta[t,s] \), then \( \gamma[t,s] \) and \( \xi[t,s',s] \) can be computed as follows

\[
\begin{align*}
\gamma[t,s] &= P(s_t = s|O,A,B) = \frac{P(s_t = s, O|A,B)}{P(O|A,B)} \tag{4.5} \\
&= \frac{\alpha[t,s] \cdot \beta[t,s]}{\sum_{s'} \alpha[t,s'] \cdot \beta[t,s']} \tag{4.6} \\
\xi[t,s',s] &= P(s_{t-1} = s',s_t = s|O,A,B) = \frac{P(s_{t-1} = s', s_t = s, O|A,B)}{P(O|A,B)} \tag{4.7} \\
&= \frac{\alpha[t-1,s] \cdot A(s',s) \cdot \beta[t,s]}{\sum_{s'} \alpha[t,s'] \cdot \beta[t,s']} \tag{4.8}
\end{align*}
\]

It only remains how to compute \( \alpha[t,s] \) and \( \beta[t,s] \). Not surprisingly, they can be computed with dynamic programming

\[
\begin{align*}
\alpha[t,s] &= \sum_{s'} \alpha[t-1,s'] \cdot A(s',s) \cdot B(s,o_t) = B(s,o_t) \sum_{s'} \alpha[t-1,s'] \cdot A(s',s) \tag{4.9}
\end{align*}
\]
\[ \beta(t,s) = \sum_{s'} \beta(t+1,s') \cdot A(s,s') \cdot B(s',o_{t+1}) \quad (4.10) \]

**M-step.** When these pseudo counts \( \gamma \) and \( \xi \) are known, then in the M-step, one can use maximum likelihood estimator to derive the updated values for \( A \) and \( B \)

\[
\hat{A}(s',s) = \frac{\text{#times } s \text{ follows } s'}{\text{#times anything follows } s'} = \frac{\sum_{t=1}^{T-1} \xi[t,s',s]}{\sum_{s''} \sum_{t=1}^{T-1} \xi[t,s',s'']} \tag{4.11}
\]

\[
\hat{B}(s,o) = \frac{\text{#times } o \text{ is observed given } s}{\text{#times anything is observed given } s} = \frac{\sum_{t=1}^{T} 1[o_t = o] \gamma[t,s]}{\sum_{t=1}^{T} \gamma[t,s]} \tag{4.12}
\]

**References**

